REMARKS

In an Office Action dated May 11, 2006, the Examiner rejected claims 1, 3, 9-10, 12 AND 20 under 35 U.S.C. §102(e) as anticipated by Hironaka, et al. (US 2004/0088489 A1); rejected claims 2 and 11 under 35 U.S.C. §103(a) as unpatentable over *Hironaka* in view of Nakamura (US 2002/0152368 A1); rejected claims 4-5, 7-8, 13-14 and 16-17 under 35 U.S.C. §103(a) as unpatentable over *Hironaka* in view of Biles (US 2004/0210749 A1); rejected claims 6 and 15 under 35 U.S.C. §103(a) as unpatentable over *Hironaka* in view of *Biles* and Rappoport, et al. (US 2005/0262332 A1); and rejected claims 18-19 under 35 U.S.C. §103(a) as unpatentable over *Hironaka* in view of Arimilli et al. (US 6,629,268 B1).

Applicant has amended all independent claims to clarify the claimed invention. In particular, the claims have been amended to clarify that a subset of multiple instructions to be executed concurrently is selected from among a potentially larger set of instructions eligible for execution, the subset being selected using the bank predict value (although other factors could be, and typically are, used as well). Other matters have also been clarified. As amended, the claims are patentable over the cited art.

A brief background discussion is in order to understand what applicant's invention is and how it works. Applicant's invention is intended for use in a processor having the capability to dispatch multiple instruction for execution in parallel in each execution dispatch cycle. Such a processor may be, but need not necessarily be, a multi-threaded processor. Preferably, a set of instructions from one or more threads is read into a buffer in the instruction unit and execution dependencies are marked in certain instructions. Instructions in the buffer, up to a marked instruction having an execution dependency, are not required by the program logic to be executed sequentially, and could therefore be executed out of order. I.e. all such instructions are eligible to execute, and any one or arbitrary subset of the set could be executed without affecting the

program logic.. The instruction unit selects instructions for dispatch from among this set of instructions. In selecting instructions for dispatch, it may take into account such factors as a thread priority or a constraint on some finite execution resource.

Loads and stores create a special problem. There can be a lot of loads and stores in an instruction stream, and it is desirable to allow multiple loads and stores to be executed concurrently. However, significant additional hardware is required for multiple read or write ports in the same cache.

In accordance with the preferred embodiment, the cache is divided into multiple banks, each having its own port (or ports) for access independently of the other banks. As long as instructions access different banks, there is no conflict and they can be dispatched simultaneously. However, because most loads and stores require that the target address be computed and/or translated, the instruction unit can not know in advance which bank will be accessed. Applicant's instruction unit therefore uses a bank predict field in the instruction to guess the cache bank in which the target data resides. It avoids dispatching two instructions which predict access to the same bank (or more instructions than the number of independent access ports to the bank, if the bank has more than one independent access port). Conflict detection circuitry will detect a conflict and restart the pipeline in the event that the instruction unit's "guess" is wrong, but it is expected that in most operating environments, the pattern of load and store accesses is sufficiently predictable that there will be a net performance improvement notwithstanding the occasional wrong guess by the instruction unit.

Therefore, a key aspect of applicant's invention is the fact that ***a bank predict field in the instruction is used by the instruction unit to select a subset of instructions, out of a potentially larger set of eligible instructions, for dispatch to the execution unit***. This critical feature is neither taught nor suggested by the cited art.

Applicant's representative claim 1, as amended, recites:

1. A digital data processing device, comprising:
instruction logic which selects and decodes instructions for execution;
execution logic which executes instructions;
a first cache for temporarily storing data, said first cache comprising a plurality of banks, each bank containing at least one respective access port for accessing data in the bank; and
wherein at least some said instructions, when executed by said execution logic, access said first cache to perform at least one of: (a) reading data from said first cache, and (b) writing data to said first cache, and wherein a respective bank predict value is associated with each of said at least some instructions accessing said first cache, each said bank predict value predicting a bank of said first cache to be accessed by its associated instruction; and
wherein said instruction logic *selects, from among a set of multiple instructions eligible to execute by said execution logic, a subset of multiple instructions for concurrent execution by said execution logic, said instruction logic using said bank predict values of said instructions to select multiple instructions which access said first cache for inclusion in said subset.* [emphasis added]

The remaining independent claims vary in scope, but all recite the key feature that a subset of instructions is selected for concurrent execution from a potentially larger set of eligible instructions by using the bank predict value.

Hironaka discloses a multi-bank integrated instruction and data cache, in which respective portions of the cache are allocated to instructions and the portions are variable to accommodate different operating environments. *Hironaka* discloses the capability to speculatively fetch multiple instructions simultaneously from the cache for possible execution, based on branch prediction logic. The Examiner apparently reads the claims on this capability.

The claims, as amended, are not anticipated by *Hironaka* for several reasons. First, the recited "bank predict value" predicts a bank of the cache ***to be accessed by its associated instruction***, which is an instruction for accessing cache. *Hironaka*'s "branch predictor" predicts a branch path of instructions, so that the processor will fetch the correct sequence of instructions.

Hironaka's "branch predictor" does not predict anything with respect to the target of any instructions which themselves access cache, i.e., load and store instructions, and is therefore not a "bank predict value" as recited in applicant's claims..

Secondly, the claims recite that the instruction logic ***selects a subset of instruction for dispatch from among a potentially larger set of instructions eligible to execute using the bank predict value***. *Hironaka*'s branch predictor selects instructions to fetch from cache. The instructions in the cache are not set of instructions which are "eligible to execute" by the execution unit, and the branch predictor is not used to select instructions for dispatch to the execution unit.

Nor are the claims obvious over *Hironaka*, either alone or in combination with the secondary references. As explained above, *Hironaka* is directed to an entirely different purpose, one of providing a variable cache width allocation between instructions and data for different operating environments. It neither discloses the existence of the problem to which applicant's invention is directed, nor suggests any particular solution. It certainly does not suggest using a bank predict field of a data access instruction to guess which bank the instruction, when executed, will access, and thus choose non-conflicting instructions for concurrent execution.

Nakamura is cited to show separate instruction and data caches, but otherwise does not teach or suggest key elements of applicant's invention. It is further observed that separate instruction and data caches of *Nakamura* are inconsistent with the entire purpose of *Hironaka*, i.e., an integrated instruction and data cache.

Biles is cited to show the use of a confirmation value. *Biles*' "confirmation signal" is used for branch prediction to confirm that a particular branch is correct. As explained above, applicant

recites a "bank prediction value", which is substantially different from branch prediction. *Biles* does not otherwise teach or suggest the key elements of applicant's invention explained above.

Rappoport is cited to show incrementing and decrementing applicant's "confirmation value". *Rappoport*, like *Biles*, relates to branch prediction and discloses incrementing and decrementing a branch prediction indicator. *Rappoport* does not otherwise teach or suggest the key elements of applicant's invention explained above.

Arimilli is cited to show multiple caches at different levels and multiple processors. In other respects, *Arimilli* relates to a test port for maintenance operations in a computer, and does not teach or suggest the key elements of applicant's invention explained above.

In view of the foregoing, applicant submits that the claims are now in condition for allowance, and respectfully requests reconsideration and allowance of all claims. In addition, the Examiner is encouraged to contact applicant's attorney by telephone if there are outstanding issues left to be resolved to place this case in condition for allowance.

Respectfully submitted,

DAVID A. LUICK

By:_____
Roy W. Truelson
Registration No. 34,265

Telephone: (507) 202-8725